Chapter 12

Assigning Proctors to Exams with Scatter Search

RAFAEL MARTÍ, HELENA LOURENÇO AND MANUEL LAGUNA
Universitat de Valencia, Universitat Pompeu Fabra and University of Colorado

Abstract: In this paper we present an algorithm to assign proctors to exams. This NP-hard problem is related to the generalized assignment problem with multiple objectives. The problem consists of assigning teaching assistants to proctor final exams at a university. We formulate this problem as an integer program (IP) with a weighted objective that combines a preference function and a workload-fairness function. We develop a scatter search procedure and compare its outcome with solutions found by solving the IP model with CPLEX 6.5. Our test problems are real instances from a University in Spain.

1. INTRODUCTION

Consider a set of teaching assistants (TAs) at a large university. Each TA has a maximum number of hours that he/she can devote to proctor final exams. This limit depends on his/her contract and teaching load. Each final exam requires a given number of TAs for proctoring. The Proctor Assignment Problem consists of assigning the TAs to the final exams. Since the TAs are graduate students, they also have final exams and therefore they cannot proctor exams during periods that conflict with their own exams. The constraints can be summarized as follows:

- Each exam must be proctored by a specified number of TAs.
- A TA cannot exceed his/her maximum number of proctor hours.
- A TA cannot proctor more than one exam at the same time.
- A TA cannot proctor a final exam that conflicts with one of his/her own.
- A TA should proctor the exams of the courses he/she taught.

Note that the last constraint can be handled before formulating the model by simply assigning proctors to the exams of the courses they taught and adjusting the associated input data accordingly (e.g., reducing the total number of proctor hours and the exam requirements). Teaching assistants have preferences for some exams, which reflect their desire for proctoring on a given day or avoiding certain days. For example, some TAs would like to avoid proctoring an exam the day before one of their exams. As a result of these preferences, one objective of the problem is to make assignments that maximize a function of the preferences.

Another important criterion that must be considered is to assign exams so the workload is evenly distributed among TA's. Unfair workloads are likely to generate conflicts among TA's and between TA's and the administration. Several objective functions can be formulated to measure the workload-fairness of a given assignment. One possibility is to minimize the difference between the TA with the largest workload and the one with the smallest. Alternatively, the model can seek to maximize the minimum workload associated with each TA. Since the number of available hours for each TA varies, the workload can be expressed as the ratio of assigned hours to available hours.

The Proctor Assignment Problem (PAP) can be viewed as an extension of the well-known Generalized Assignment Problem (GAP) (see Lourenço and Serra 1998; Laguna, et al. 1995; Osman 1995; and Chu and Beasley 1997), by allowing more than one agent to be assigned to a task and introducing side constraints. The side constraints model aspects of the problem that are not part of the GAP, such as the need for assigning multiple proctors to an exam and for avoiding the assignment of the same TA to multiple exams that are held at the same time period. Moreover, instead of a single objective function that maximizes the total profit in the GAP, the PAP considers two objective functions to simultaneously maximize preferences and fairness. Therefore, the PAP can be formulated as multi-objective integer program. We propose a heuristic procedure based on the scatter search methodology to find solutions to the PAP. The procedure will be embedded in a user-friendly system to help the planning of final exams in the School of Economics of the University Pompeu Fabra at Barcelona (Spain).

2. PROBLEM FORMULATION

In order to simplify the problem and mimic the manual method currently practiced at the University Pompeu Fabra of Barcelona, we split an exam day into two periods (8:00 AM - 2:00 PM and 2:00 PM - 9:00 PM). Then if d is the total number of exam days, $k = 2d$ is the total number of periods. The following assumptions are made:

- An exam starts and ends in the same day.
- An exam can be scheduled over one or two periods in a given day.
- TA's express their preferences for given periods.

In order to standardize the TA preferences, we translate the preferences for periods to preferences for exams. That is, if a TA expresses a strong preference for period 1 (8:00 AM - 2:00 PM in the first day of final exams), then we consider that the TA has a strong preference for all exams scheduled during this period. If an exam is scheduled over two periods (e.g., an exam that starts at noon and finishes at 3:00 PM on the first day of final exams), then we use the lower value between the preference for period 1 and 2, as expressed by each TA.

Let J be the set of m exams (j = 1, ..., m) and I the set of n TA's (i = 1, ..., n). Then, the following notation is used to represent the relevant data in our problem:

a_i = maximum number of available hours for TA i.
b_j = number of hours associated with exam j .
t_j = number of TA's required for exam j.
c_{ij} = preference of TA i for the exam j.
P_i = the set of exams that overlap with any TA i's exams.
T_k = the set of exams scheduled in period k.
d = number of exam days.

Also, we define the set of binary variables x_{ij}, such that x_{ij} = 1 if TA i is assigned to exam j; and x_{ij} = 0 otherwise. Using these definitions, the PAP can be formulated as follows:

$$\text{Max} \quad f(x) = \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij} \tag{1}$$

$$\text{Min} \quad g(x) = y \tag{2}$$

$$\text{Subject to} \quad \sum_{j=1}^{m} b_j x_{ij} \le a_i \qquad i = 1, \ldots, n \tag{3}$$

$$\sum_{j=1}^{m} b_j x_{ij} - a_i y \geq 0 \qquad i = 1, \ldots, n \qquad (4)$$

$$\sum_{i=1}^{n} x_{ij} = t_j \qquad j = 1, \ldots, m \qquad (5)$$

$$\sum_{i \in T_k}^{n} x_{ij} \leq 1 \qquad i = 1, \ldots, n;\ k = 1, \ldots, 2d \qquad (6)$$

$$x_{ij} = \{0,1\} \qquad i = 1, \ldots, n;\ j = 1, \ldots, m \qquad (7)$$

$$x_{ij} = 0 \qquad j \in P_i \qquad (8)$$

$$y \geq 0 \qquad (9)$$

Equation (1) and (2) represent the sum of the preferences and the minimum TA utilization, respectively. These objectives are to be maximized. The utilization is the fraction of the available hours that a TA is assigned to proctor exams. Constraint (3) limits the number of assigned hours to not exceed the available hours for each TA. Constraint (4) calculates the minimum TA utilization. Constraint (5) guarantees that each exam has the required number of TA's. Constraint (6) guarantees that each TA can proctor at most one exam in the same period. Constraint (7) enforces the binary restrictions on the decision variables. Constraint (8) eliminates the assignments that create a conflict with the exams that proctors have. Finally, constraint (9) enforces the nonnegativity restriction on the y-variable.

We combine the objective functions to create a single, weighted, function of the following form:

$$h(x) = f'(x) + a g(x) \qquad (10)$$

where

$$f'(x) = \frac{\sum_{i=1}^{n} \sum_{j=1}^{m} \frac{c_{ij}}{c_{\max}(j)} x_{ij}}{\sum_{j=1}^{m} t_j} \qquad (11)$$

and

$$c_{\max}(j) = \max_i (1, c_{ij}). \qquad (12)$$

Note that since the minimum preference value is zero, then both $f'(x)$ and $g(x)$ are bounded between 0 and 1. The value of $a \geq 0$ is used to show the preference of the decision maker towards either element of the combined objective function. If $a > 1$, then the decision maker prefers assignments with a more uniform distribution of the workload. If $a < 1$, then the decision maker prefers assignments that maximize the total (normalized) preference value.

3. SCATTER SEARCH APPROACH

Scatter search, from the standpoint of metaheuristic classification, may be viewed as an evolutionary (or also called population-based) algorithm that constructs solutions by combining others. It derives its foundations from strategies originally proposed for combining decision rules and constraints (in the context of integer programming). The goal of this methodology is to enable the implementation of solution procedures that can derive new solutions from combined elements in order to yield better solutions than those procedures that base their combinations only on a set of original elements. As described in tutorial articles (Glover 1998 and Laguna 1999) and other implementations based on this framework (Campos et al. 1998), the methodology includes the following basic elements:

- Generate a population *P*.
- Extract a reference set *R*.
- Combine elements of *R* and maintain and update *R*.

Scatter search finds improves solutions by combining solutions in *R*. This set, known as the reference set, consists of solutions of high quality that are also diverse. The overall proposed procedure, based on the scatter-search elements listed above, follows:

Preprocessing: Exams with a small number of students are proctor by the course's professor. TA's are assigned to the exams corresponding to the courses they taught. Exam and TA data are updated to reflect these assignments.

Generate a population P: Apply the diversification generator to generate diverse solutions.

Construct the reference set R: Add to R the best r_1 solutions in P. Add also r_2 diverse solutions from P to construct R with $|R| = r_1 + r_2$ solutions.

Maintain and update the reference ret R: Apply the subset generation method (Glover 1998) to combine solutions from R. Update R, adding solutions that improve the quality of the worst in the set.

We now describe the implementation details of the main elements of the procedure, as adapted in the context of the exam proctor assignment.

3.1 Diversification Generation Method

An initial population of solutions is constructed by means of a diversification generator. The generator that we implemented is based on the notion of constructing solutions employing modified frequencies. The goal of the method is to generate good-quality diverse solutions. The generator uses the following frequency function:

$$f_{ij} = \sum_{x \in P} x_{ij} \tag{13}$$

This frequency value is used to bias the potential assignment of TA i to exam j during subsequent constructions of solutions, and therefore to induce diversity in the new solutions with respect to the solutions already in P.

The attractiveness of assigning TA to an exam is given by his/her preference. The preference of TA i for exam j (c_{ij}) is a value in the range [0,5]. The preference value is computed as the difference between the period in which exam j is scheduled and the period of the closest exam that TA i must take. Differences of more than 5 periods are adjusted back to 5. The period just before one for which a TA has an exam is assigned a preference of 0.

We modify the value of c_{ij} to reflect previous assignments of TA i to exam j, as follows:

$$c'_{ij} = c_{ij} - b\left(\frac{\max_{i,j} \quad c_{ij}}{\max_{i,j} \quad f_{ij}}\right) f_{ij} \tag{14}$$

It should be noted that c'_{ij} is an adaptive function, since its value depends on attributes of the unassigned elements and a function of previous assignments. The value of b is dynamically modified to encourage additional diversification. If the generator constructs the same solution more than once, the b -value is increased. In our implementation, we start with $b = 0.4$ and increase its value by 0.1 every time the generator repeats a construction.

Figure 1 summarizes the diversification generation method. The method generates *PopSize* solutions using the updated c'_{ij} values. TA's are assigned to exams in order to maximize the modified preference values. The procedure stops when *PopSize* solutions are generated.

Initialization
    Solutions = 0;
    f_{ij}=0 for all i,j

While (*Solutions* < *PopSize*)
{
    For k=1 to 2d
    {
        Order the exams in period k by decreasing number of required TA's.
        For each exam j in period k
        {
            Construct the list of TA's that can proctor j.
            Order the list according to c_{ij}
            Assign the first t_j TA's to proctor exam j.
            Update TA and exam information.
        }
    }
    Add the solution to the population.
    Make *Solutions* = *Solutions* + 1;
    Update the corresponding f_{ij}.
}

Figure 1. Diversification generator

3.2 Updating and Maintaining the Reference Set

The reference set R is a subset of the population set P that consists of high quality and diverse solutions. A distance function $\partial(x', x'')$, between two solutions x' and x'', is used to measure the diversity of the solutions in the reference set.

$$\partial(x', x'') = \sum_{i=1}^{n} \sum_{j=1}^{m} \left| x'_{ij} - x''_{ij} \right| \tag{15}$$

Note that a large distance between two solutions does not translate into a large difference between their corresponding objective function values. This is why diversity of the solutions in *R* cannot be measure with reference to the objective function values only.

The reference set *R* with $|R| = b$ is constructed as follows. Order the solutions in *P* by decreasing value of the objective function. Select the first $b/2$ solutions and add them to *R*. For each solution x in *P*-*R*, calculate the distance to all the elements in *R*. Let the minimum distance $\partial_{\min}(x)$ from a solution x in *P*-*R* to all solutions x' in *R* be defined as:

$$\partial_{\min}(x) = \min_{x' \in R} \{\partial(x, x')\} \tag{16}$$

Select the solution x^* with the maximum distance $\partial_{\min}(x)$ of all x in *P*-*R*. Add x^* to *R*, until $|R| = b$. In our experiments, we use $b = 20$, because previous studies (Campos, et al. 1998 and 1999) have shown the merit of such choice.

3.3 Solution Combination Method

Scatter search generates new solutions by combining those in the reference set. Specifically, a combination method is applied to subsets of solutions in the reference set. A newly generated solution is compared with the worst solution in *R*. The new solution replaces the worst solution if the objective value of the new solution is better than the objective value of the worst solution in *R*.

The solution combination procedure seeks to generate subsets *X* of *R* that have useful properties, while avoiding the duplication of subsets previously generated. The approach for doing this is organized to generate four different collections of subsets of *R*, which Glover (1998) refers to as *SubSetType* 1, 2, 3 and 4:

SubsetType 1: all 2-element subsets.

SubsetType 2: 3-element subsets derived from the 2-element subsets by augmenting each 2-element subset to include the best solution not in this subset.

SubsetType 3: 4-element subsets derived from the 3-element subsets by augmenting each 3-element subset to include the best solutions not in this subset.

SubsetType 4: the subsets consisting of the best i elements, for $i = 5$ to b.

A central consideration of this design is that R itself might not be static, because it might be changing as new solutions are added to replace old ones (when these new solutions qualify to be among the current b best solutions found).

The solution combination method is applied to each subset. This is based on a voting system, where each solution votes for specific assignments of TA's to exams. The resulting construction may be infeasible with respect to constraints (3) and (6), in which case, a repair mechanism is applied. The voting scheme is summarized in Figure 2.

```
For each exam j
{
        Find the assignment x_ij with the largest vote.
        Assign the exam j to TA i.
}
```

Figure 2. Voting mechanism

The vote associated with the assignments $x_{\bullet j}$ is a measure of the merit of assigning TA's to exam j. The vote is therefore defined to take into account the preference of the TA's for some exams. In addition, the vote penalizes assignments that result in violations of one or more of the problem constraints.

The process consists of m steps (where m is the number of exams). At each step j, a solution x votes for its column $x_{\bullet j}$, that consists of all the TA assignments associated with exam j. The vote of solution x at step j is calculated as follows:

$$V_j(x) = \sum_{i=1}^{n} \left(\begin{array}{l} c_{ij}x_{ij} - j\left(\max\left(0, \sum_{l=1}^{j-1} b_l y_{il} + b_j x_{ij} - a_i \right)\right) \\ -g\left(\max\left(0, \sum_{k:j\in T_k} \left(\sum_{l\in T_k, l\le j-1} y_{il} + x_{ij} - 1 \right)\right)\right) \end{array} \right) \tag{17}$$

The first term of the vote calculation adds the preferences of the TA's assigned to exam j in solution x. The second term calculates the number of hours assigned to each TA in the solution that is being constructed (represented by the y-variable). Then, it adds the number of hours for the current exam j and then subtracts the available hours for the TA. If the hours already assigned plus the hours related to the current exam exceed the available number of hours, then the excess hours are multiplied by a penalty factor φ. In other words, the second term in equation (17) penalizes violations of constraint (3). In a similar way, the third term of equation (1) penalizes the violation of constraint (6), by multiplying the number of times a TA is assigned to proctor different exams in the same time period by the constant γ.

When the combination method results in a solution that violates either constraint (3) or constraint (6), the procedure in Figure 3 is executed in an attempt to repair the newly generated solution.

```
For each period k
{
        For each exam j scheduled in period k
        {
                If any TA assigned to j is also assigned to
                another exam in period k or his/her capacity
                has been exceeded
                {
                        Find the next available TA that can
                        be feasibly assigned to exam j. Stop
                        if no such TA can be found.
                }
        }
}
```

Figure 3. Solution repair procedure

Although the procedure in Figure 3 may fail to repair a solution, our experience has been that the method seldom fails.

4. COMPUTATIONAL EXPERIMENTS

The data used for these experiments correspond to real instances of the proctor assignment problem at the Universitat Pompeu Fabra in Barcelona (Spain). Presently, proctors are manually assigned to exams, following simple rules that enforce the constraints in the problem. We compare our results with assignments that were generated manually and also with assignments found solving the mixed-integer programming formulation with Cplex 6.5 (some of which are optimal).

Our test set consists of 11 problems. The results are summarized in Table 1. The first three columns in this table show the problem number, the number of TA's and the number of exams, respectively. The table than shows the objective function value corresponding to the scatter search solution and the solution found with Cplex. The asterisk in the Cplex column indicates that the solution was confirmed to be optimal.

Table 1 shows that scatter search solutions are inferior to those found by Cplex in the set of problems used for testing and the objective function chosen to measure performance. However, an advantage of scatter search is that the reference set contains a number of high-quality solutions from which the decision-maker could choose the one to implement. The maximum standard deviation of the objective function value for solutions in the final reference set was 0.000407 for all problem instances. This indicates that practically all of the solutions in the final reference set have the same quality with respect to the objective function value. Since the objective function is a mathematical representation of some subjective measure of performance associated with a given assignment, the ability to choose among solutions that have similar objective function values is an important feature of a solution procedure to be embedded in a decision support system designed for this managerial situation.

Table 1. Summary of results

Problem	TA's	Exams	Scatter Search	Cplex (* = optimal)
1	23	21	0.764	0.953
2	59	52	1.000	1.393
3	59	44	1.000	1.165
4	25	21	0.754	0.815
5	46	42	0.952	1.268
6	92	84	1.000	1.308
7	139	125	1.000	1.371
8	139	125	1.000	1.000*
9	139	125	1.000	1.000*
10	166	182	0.999	1.000*
11	180	210	0.116	1.000*

The scatter search procedure was coded in C and run on a Pentium II computer at 350 MHz. The computational times are very reasonable, with an average of 102.3 seconds and a maximum of 208.2 seconds. Cplex 6.5 was also run on a Pentium II machine at 350 MHz. The depth-first search strategy was used with a time limit of one hour.

5. CONCLUSIONS

One of the most interesting features of the proctor assignment problem is its multi-objective nature. In the current development, we have formulated the problem as a mixed-integer program with a single objective and implemented a scatter search solution procedure. We anticipate that this procedure can be the basis for one that directly exploits the multi-objective characteristics of this problem. This generalization can be achieved by constructing and updating a reference set R consisting of non-dominated solutions.

A solution x is dominated if it exits another solution x' in R such that $f(x') > f(x)$ and $g(x') < g(x)$. In other words, since the objectives of the problem are to maximize $f(x)$ and minimize $g(x)$, a solution x is dominated when there is a solution in x' that is better than x with respect to both objectives. This definition would create a dynamic update of R, where the cardinality of the set would vary with the set of non-dominated solutions. The implementation of a procedure that uses these definitions and is based on the mechanisms developed in this paper will be the topic of a future project.

REFERENCES

Awad, R. and J. Chinneck (1998) "Proctor Assignment at Carleton University," *Interfaces,* vol. 28, no. 2, pp. 58-71.

Campos, V., F. Glover, M. Laguna and R. Martí (1999) "An Experimental Evaluation of a Scatter Search for the Linear Ordering Problem," University of Colorado at Boulder.

Campos, V., M. Laguna and R. Martí (1998) "Scatter Search for the Linear Ordering Problem," to appear in *New Methods in Optimisation,* D. Corne, M. Dorigo and F. Glover (Eds.), McGraw-Hill.

Chu, P. C. and J. E. Beasley (1997) "A Genetic Algorithm for the Generalized Assignment Problem," *Computers and Operations Research*, vol. 24, pp. 17-23.

Glover, F. (1998) "A Template for Scatter Search and Path Relinking," in *Artificial Evolution, Lecture Notes in Computer Science* 1363, J.-K. Hao, E. Lutton, E. Ronald, M. Schoenauer and D. Snyers (Eds.), Springer-Verlag, pp. 13-54.

Laguna, M., J. P. Kelly, J. L. González Velarde and F. Glover (1995) "Tabu Search for the Multilevel Generalized Assignment Problem," *European Journal of Operational Research*, vol. 82, pp. 176-189.

Lourenco HR and Serra D, 1998, Adaptive approach heuristics for the generalized assignment problem', DEE-UPF, *Economic Working paper Series* No. 288, May.

Osman, I. H. (1995) "Heuristics for the Generalized Assignment Problem: Simulated Annealing and Tabu Search Approaches," *OR Spektrum*, vol. 17, pp. 211-225.